INT353

Name:	IMMIDICHETTY REDDY SWETHAK
Reg No:	12008336
Section:	K20PM
Roll No:	RK20PMA18
Group:	B.Tech(CSE)

EXPLORATORY DATA ANALYSIS REPORT

My Dataset

NBA games data

(from 2004 season to dec 2020)

INTRODUCTION

This dataset was selected to work on the NBA games data. I taken the data from the nba stats website to create this dataset. In this dataset the data is consists from the 2004 season to December 2020 season.

This dataset is about the National Basketball Association. Perfect for machine learning / sports data analysis & visualization, and building sportsbetting prediction models. NBA Player List (CSV) Data for every player to have ever played in the NBA, and each player’s player id.

CONTENT

You can find 5 datasets :

- games.csv : all games from 2004 season to last update with the date, teams and some details like number of points, etc.
- games_details.csv : details of games dataset, all statistics of players for a given game
- players.csv : players details (name)
- ranking.csv : ranking of NBA given a day (split into west and east on `CONFERENCE` column
- teams.csv : all teams of NBA

DOMAIN

Games and sports are very important in one’s life.Those who participate in games and sports have a good outlook on life, because they are likely to be physically and mentally fit. They are beneficial in a variety of ways, including helping to maintain blood pressure, increase blood flow, improve thinking capacity and attention, and so on.

It assists in the development of a team spirit and develops a leadership quality in the individual, in addition to being physically and intellectually fit. When people participate in sports or games, they become more intelligent, energetic, and courageous. Many children pursue careers in numerous sports and games, making them well-known figures in society.

DETAILS

The dataset refers to the statistics of 1749 playes are from the 2004 season to December 2020 season. In the dataset having Game's date, ID of the game, Status : Final means that the is completed, ID of the home team, ID of the visitor team, Season when the game occurred, ID of the home team, Number of points scored by home team, Field Goal Percentage home team, Free Throw Percentage of the home team.

Why?

I have taken this dataset because it is a based on the bakestball dataset and I know maximum about basketball players and it is also contains the different plays and I will also know many players.

I am so much interested in working with this dataset and so I have taken this type of dataset and I have so much interested to study this type of dataset and to analysis the dataset and to perform visualization of this dataset

Question/Plans

- Displaying the basic statistical details of the data.
- Removing null values
- Cleaning the data by replacing missing values, repeated values.
- Visualization of data by using matplotlib.
- Finding the outliers.
- Eliminating the outliers.
- Dropping unwanted columns.
- Finding relation between the columns.
- Plotting the bar chart.
- Checking the density of particular column by using KDE plot.
- Visualization of correlation graph by using heatmap.
- Perform Univariate Analysis and Bivariate Analysis.

Acknowledgements

I would like to thanks NBA stats website which allows all NBA data freely open to everyone.

Displaying the dataset:

```
#Reading csv file
df = pd.read_csv("C:/Users/swethak/Desktop/EDA project/games.csv")
```

```
# Display the data set
df
```

	GAME_DATE_EST	GAME_ID	GAME_STATUS_TEXT	HOME_TEAM_ID	VISITOR_TEAM_ID	SEASON	TEAM_ID_home	PTS_home	FG_PCT_home	FT_
0	2022-03-12	22101005	Final	1610612748	1610612750	2021	1610612748	104.0	0.398	
1	2022-03-12	22101006	Final	1610612741	1610612739	2021	1610612741	101.0	0.443	
2	2022-03-12	22101007	Final	1610612759	1610612754	2021	1610612759	108.0	0.412	
3	2022-03-12	22101008	Final	1610612744	1610612749	2021	1610612744	122.0	0.484	
4	2022-03-12	22101009	Final	1610612743	1610612761	2021	1610612743	115.0	0.551	
...	...	...	...	...	...	...	...	...	...	
25791	2014-10-06	11400007	Final	1610612737	1610612740	2014	1610612737	93.0	0.419	
25792	2014-10-06	11400004	Final	1610612741	1610612764	2014	1610612741	81.0	0.338	
25793	2014-10-06	11400005	Final	1610612747	1610612743	2014	1610612747	98.0	0.448	
25794	2014-10-05	11400002	Final	1610612761	1610612758	2014	1610612761	99.0	0.440	
25795	2014-10-04	11400001	Final	1610612748	1610612740	2014	1610612748	86.0	0.431	

25796 rows × 21 columns

Head and Tail of the dataset:

```
# Display first 5 rows
df.head()
```

	GAME_DATE_EST	GAME_ID	GAME_STATUS_TEXT	HOME_TEAM_ID	VISITOR_TEAM_ID	SEASON	TEAM_ID_home	PTS_home	FG_PCT_home	FT_PCT_
0	2022-03-12	22101005	Final	1610612748	1610612750	2021	1610612748	104.0	0.398	
1	2022-03-12	22101006	Final	1610612741	1610612739	2021	1610612741	101.0	0.443	
2	2022-03-12	22101007	Final	1610612759	1610612754	2021	1610612759	108.0	0.412	
3	2022-03-12	22101008	Final	1610612744	1610612749	2021	1610612744	122.0	0.484	
4	2022-03-12	22101009	Final	1610612743	1610612761	2021	1610612743	115.0	0.551	

5 rows × 21 columns

```
# Display last 5 rows
df.tail()
```

	GAME_DATE_EST	GAME_ID	GAME_STATUS_TEXT	HOME_TEAM_ID	VISITOR_TEAM_ID	SEASON	TEAM_ID_home	PTS_home	FG_PCT_home	FT_
25791	2014-10-06	11400007	Final	1610612737	1610612740	2014	1610612737	93.0	0.419	
25792	2014-10-06	11400004	Final	1610612741	1610612764	2014	1610612741	81.0	0.338	
25793	2014-10-06	11400005	Final	1610612747	1610612743	2014	1610612747	98.0	0.448	
25794	2014-10-05	11400002	Final	1610612761	1610612758	2014	1610612761	99.0	0.440	
25795	2014-10-04	11400001	Final	1610612748	1610612740	2014	1610612748	86.0	0.431	

5 rows × 21 columns

Shape,size and describe of the dataset:

```
1 # Shape of the dataset
2 df.shape
```

(25796, 21)

```
1 #Size of the dataset
2 df.size
```

541716

Finding the “dtypes” of columns in the dataset:

```
1 # Information about the dataset
2 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25796 entries, 0 to 25795
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   GAME_DATE_EST    25796 non-null  object 
 1   GAME_ID          25796 non-null  int64  
 2   GAME_STATUS_TEXT 25796 non-null  object 
 3   HOME_TEAM_ID     25796 non-null  int64  
 4   VISITOR_TEAM_ID  25796 non-null  int64  
 5   SEASON           25796 non-null  int64  
 6   TEAM_ID_home     25796 non-null  int64  
 7   PTS_home         25697 non-null  float64
 8   FG_PCT_home      25697 non-null  float64
 9   FT_PCT_home      25697 non-null  float64
 10  FG3_PCT_home     25697 non-null  float64
 11  AST_home         25697 non-null  float64
 12  REB_home         25697 non-null  float64
 13  TEAM_ID_away     25796 non-null  int64  
 14  PTS_away         25697 non-null  float64
 15  FG_PCT_away      25697 non-null  float64
 16  FT_PCT_away      25697 non-null  float64
 17  FG3_PCT_away     25697 non-null  float64
 18  AST_away         25697 non-null  float64
 19  REB_away         25697 non-null  float64
 20  HOME_TEAM_WINS   25796 non-null  int64  
dtypes: float64(12), int64(7), object(2)
memory usage: 4.1+ MB
```

Numeric features:

```
1 #Describe the dataset
2 df.describe()
```

	GAME_ID	HOME_TEAM_ID	VISITOR_TEAM_ID	SEASON	TEAM_ID_home	PTS_home	FG_PCT_home	FT_PCT_home	FG3_PCT_home	
count	2.579600e+04	2.579600e+04	2.579600e+04	25796.000000	2.579600e+04	25697.000000	25697.000000	25697.000000	25697.000000	25
mean	2.169208e+07	1.610613e+09	1.610613e+09	2011.798341	1.610613e+09	103.106044	0.460313	0.759705	0.355896	
std	5.496041e+06	8.638857e+00	8.654846e+00	5.397985	8.638857e+00	13.174726	0.056629	0.100692	0.111940	
min	1.030000e+07	1.610613e+09	1.610613e+09	2003.000000	1.610613e+09	36.000000	0.250000	0.143000	0.000000	
25%	2.060109e+07	1.610613e+09	1.610613e+09	2007.000000	1.610613e+09	94.000000	0.421000	0.696000	0.286000	
50%	2.120040e+07	1.610613e+09	1.610613e+09	2012.000000	1.610613e+09	103.000000	0.459000	0.765000	0.355000	
75%	2.170070e+07	1.610613e+09	1.610613e+09	2016.000000	1.610613e+09	112.000000	0.500000	0.829000	0.429000	
max	5.200021e+07	1.610613e+09	1.610613e+09	2021.000000	1.610613e+09	168.000000	0.684000	1.000000	1.000000	

Checking the null values:

```
#checking the null values
df.isnull().sum()
```

```
GAME_DATE_EST        0
GAME_ID              0
GAME_STATUS_TEXT     0
HOME_TEAM_ID         0
VISITOR_TEAM_ID      0
SEASON               0
TEAM_ID_home         0
PTS_home            99
FG_PCT_home         99
FT_PCT_home         99
FG3_PCT_home        99
AST_home            99
REB_home            99
TEAM_ID_away         0
PTS_away            99
FG_PCT_away         99
FT_PCT_away         99
FG3_PCT_away        99
AST_away            99
REB_away            99
HOME_TEAM_WINS       0
dtype: int64
```

Removing the null values:

```
GAME_DATE_EST       0
GAME_ID             0
GAME_STATUS_TEXT    0
HOME_TEAM_ID        0
VISITOR_TEAM_ID     0
SEASON              0
TEAM_ID_home        0
PTS_home            0
FG_PCT_home         0
FT_PCT_home         0
FG3_PCT_home        0
AST_home            0
REB_home            0
TEAM_ID_away        0
PTS_away            0
FG_PCT_away         0
FT_PCT_away         0
FG3_PCT_away        0
AST_away            0
REB_away            0
HOME_TEAM_WINS      0
dtype: int64
```

Replacing the missing values:

```
#replacing missing values
nr=df['PTS_home'].replace(np.NaN,df['PTS_home'].median(),inplace=True)
nr
```

```
#checking the null values
df.isnull().sum()
```

```
GAME_DATE_EST        0
GAME_ID              0
GAME_STATUS_TEXT     0
HOME_TEAM_ID         0
VISITOR_TEAM_ID      0
SEASON               0
TEAM_ID_home         0
PTS_home             0
FG_PCT_home         99
FT_PCT_home         99
FG3_PCT_home        99
AST_home            99
REB_home            99
TEAM_ID_away         0
PTS_away            99
FG_PCT_away         99
FT_PCT_away         99
FG3_PCT_away        99
AST_away            99
```

Outliers:

Detecting position of the outliers:

```
#detecting position of outliers
print(np.where(df['FG3_PCT_away']>0))
```

```
(array([    0,     1,     2, ..., 25694, 25695, 25696], dtype=int64),)
```

Detecting of outliers using z-score method:

```
#detection of outliers using z-zscore method
from scipy import stats
import numpy as np
z=np.abs(stats.zscore(df['FG3_PCT_away']))
print(z)
```

```
0        0.068852
1        1.283337
2        0.359255
3        0.332029
4        0.341104
           ...
25791    0.232203
25792    0.747907
25793    1.366590
25794    0.322954
25795    0.803934
Name: FG3_PCT_away, Length: 25697, dtype: float64
```

Detecting outliers using IQR method:

```
#detecting outliers using IQR method
Q1=np.percentile(df['FG3_PCT_away'],25,interpolation = 'midpoint')
Q3=np.percentile(df['FG3_PCT_away'],75,interpolation = 'midpoint')
IQR = Q3-Q1
```

```
upper = df['FG3_PCT_away'] >=(Q3+1.5*IQR)
print("Upper bound:",upper)
print(np.where(upper))

lower = df['FG3_PCT_away'] <= (Q1-1.5*IQR)
print("Lower bound:",lower)
print(np.where(lower))
```

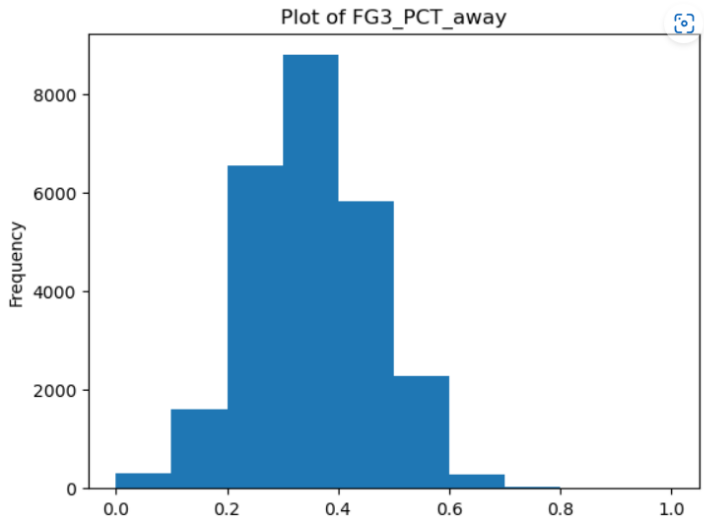
```
Upper bound: 0        False
1        False
2        False
3        False
4        False
         ...
25791    False
25792    False
25793    False
25794    False
25795    False
Name: FG3_PCT_away, Length: 25697, dtype: bool
(array([ 1298,  2607,  3898,  3961,  4499,  4554,  4623,  5162,  5276,
         5452,  6081,  6268,  6381,  6440,  6512,  6563,  6672,  6904,
         7034,  7497,  8110,  8121,  8136,  8185,  8229,  8285,  8314,
         8357,  8399,  8424,  8434,  8568,  8581,  8760,  8777,  8950,
         8997,  9094,  9157,  9185,  9203,  9244,  9353,  9403,  9671,
         9775,  9788,  9831,  9913,  9943,  9954,  9968, 10140, 10227,
        10242, 10334, 10421, 10510, 10563, 10593, 10743, 10766, 10854,
        10940, 11043, 11120, 11149, 11177, 11265, 11386, 11445, 11734,
        11797, 11855, 11915, 11954, 11966, 11971, 11987, 12005, 12085,
        12123, 12186, 12193, 12254, 12646, 12765, 13013, 13031, 13145,
        13285, 13364, 13430, 13544, 13547, 13698, 13718, 13729, 13744,
        13772, 13879, 13882, 14089, 14116, 14519, 14562, 14618, 14678,
        14689, 14703, 14745, 14753, 14761, 14792, 14818, 14833, 14864,
        14892, 14922, 14980, 14983, 15018, 15162, 15201, 15252, 15272,
        15281, 15282, 15325, 15339, 15353, 15367, 15485, 15514, 15680,
        15882, 15893, 15937, 16051, 16098, 16130, 16162, 16171, 16216,
        16228, 16271, 16314, 16362, 16380, 16442, 16468, 16586, 16620,
        16710, 16842, 16851, 16883, 16967, 17229, 17251, 17337, 17366,
        17493, 17569, 17662, 17663, 17669, 17684, 17700, 17706, 17751,
        17814, 17849, 17921, 17992, 18008, 18045, 18070, 18075, 18168,
        18214, 18259, 18323, 18327, 18610, 19207, 19522, 20352, 20595,
        20857, 21731, 22485, 22899, 23226, 23539, 23965, 24068, 24945,
        25208, 25244], dtype=int64),)
```

```
Lower bound: 0        False
1        False
2        False
3        False
4        False
         ...
25791    False
25792    False
25793    False
25794    False
25795    False
Name: FG3_PCT_away, Length: 25697, dtype: bool
(array([ 3337,  3487,  4525,  4884,  4925,  5347,  5567,  5609,  6128,
         6198,  6509,  6526,  6544,  6634,  6841,  6873,  7016,  7028,
         7030,  7066,  7116,  7127,  7596,  7699,  7854,  7856,  8286,
         8385,  8419,  8515,  8867,  8922,  8981,  9098,  9152,  9267,
         9283,  9418,  9438,  9769,  9875,  9878, 10105, 10232, 10333,
        10686, 10899, 11273, 11285, 11308, 11414, 11496, 11800, 11904,
        11936, 12002, 12010, 12214, 12408, 12413, 12568, 12680, 12720,
        12787, 12795, 12953, 12973, 13052, 13074, 13137, 13209, 13245,
        13272, 13713, 14006, 14027, 14033, 14046, 14158, 14220, 14274,
        14634, 14730, 14795, 14803, 14851, 14889, 15129, 15208, 15215,
        15247, 15307, 15528, 15557, 15572, 15579, 15596, 15608, 15609,
        15626, 15656, 15692, 15857, 15884, 16016, 16351, 16365, 16381,
        16429, 16533, 16558, 16657, 16658, 16679, 16728, 16816, 16826,
        16895, 16923, 16970, 17037, 17038, 17083, 17247, 17267, 17320,
        17325, 17330, 17495, 17876, 17985, 17998, 18023, 18182, 18297,
        18324, 22525, 22832, 22933, 23422, 23486, 23865, 24885, 25183,
        25455, 25559], dtype=int64),)
```

Univariate Analysis:

Plot:

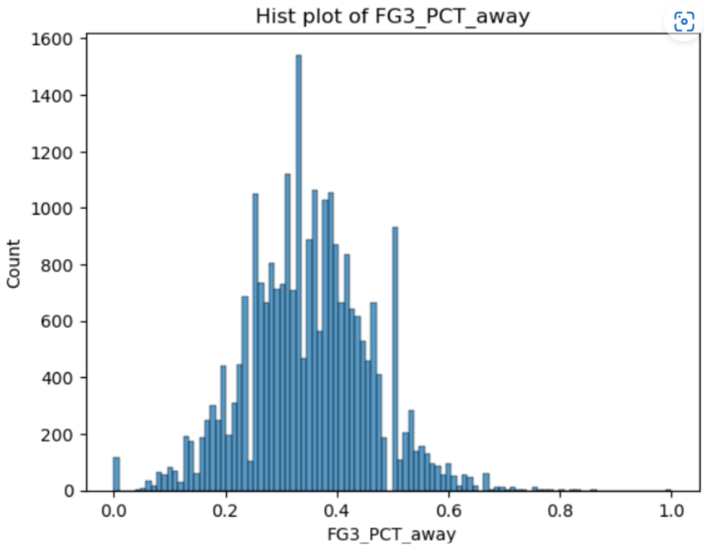
```
df.FG3_PCT_away.plot.hist()
plt.title("Plot of FG3_PCT_away")
plt.show()
```



1.Hist plot:

```
sns.histplot(df['FG3_PCT_away'])
plt.title("Hist plot of FG3_PCT_away")
```

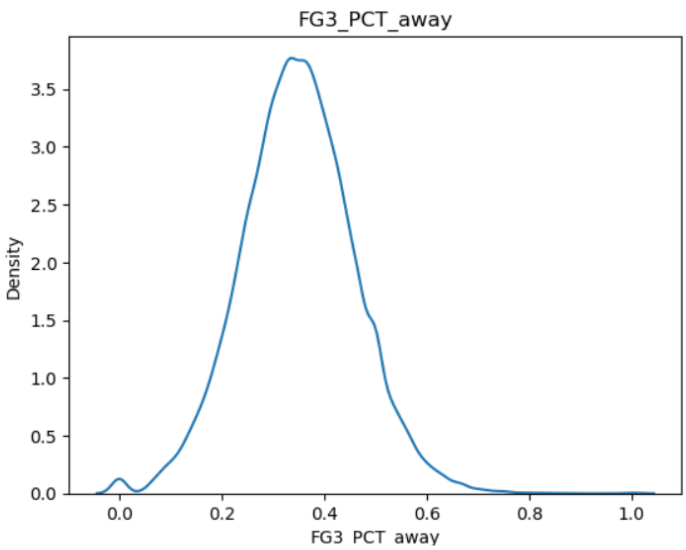
```
Text(0.5, 1.0, 'Hist plot of FG3_PCT_away')
```



2.Kernal Density plot:

```
sns.distplot(df['FG3_PCT_away'],hist=False)
plt.title("FG3_PCT_away")
plt.figure(figsize=(20,14))
plt.show()
```

```
C:\Users\swethak\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated functi
on and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with sim
ilar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```

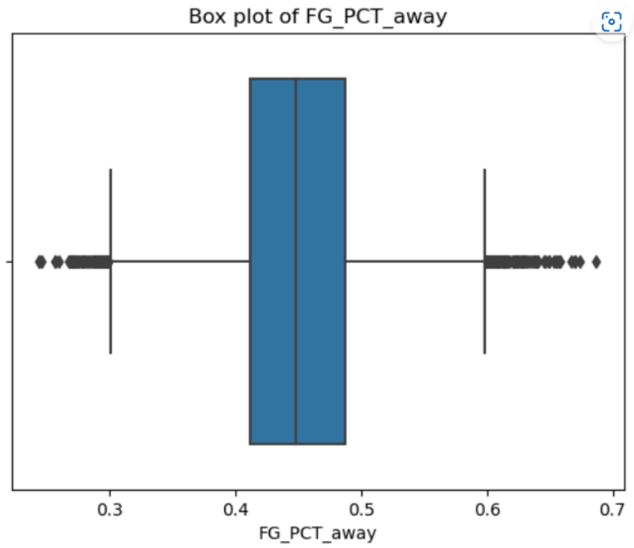


3.Box plot:

```
sns.boxplot(df.FG_PCT_away)
plt.title("Box plot of FG_PCT_away")
plt.show
```

```
C:\Users\swethak\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyw
ord arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an exp
licit keyword will result in an error or misinterpretation.
  warnings.warn(
```

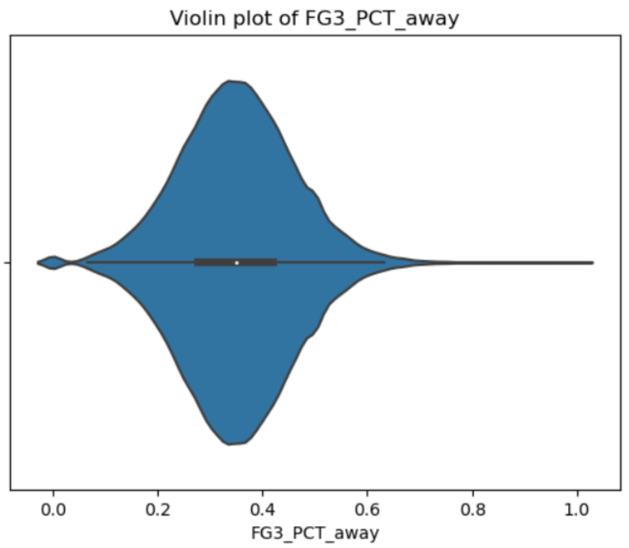
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



4.Violin plot:

```
sns.violinplot(df["FG3_PCT_away"])
plt.title("Violin plot of FG3_PCT_away")
plt.show()
```

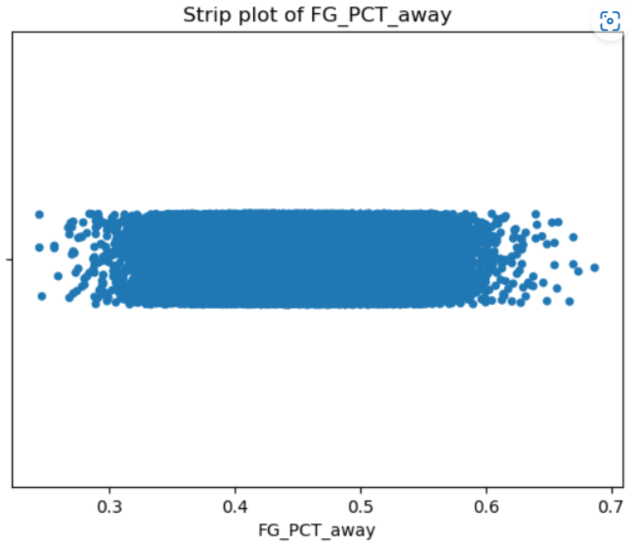
```
C:\Users\swethak\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyw
ord arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an exp
licit keyword will result in an error or misinterpretation.
  warnings.warn(
```



5.Strip plot:

```
sns.stripplot(df["FG_PCT_away"])
plt.title("Strip plot of FG_PCT_away")
plt.show()
```

```
C:\Users\swethak\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyw
ord arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an exp
licit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Barplot for the dataset:

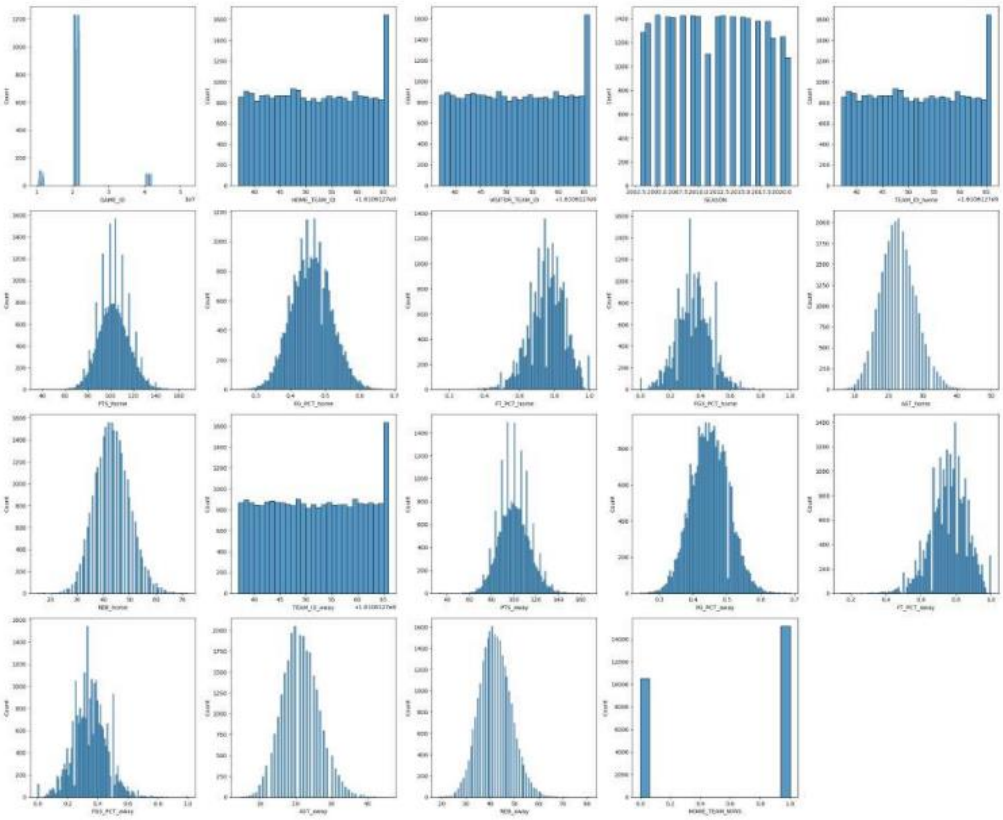
```
#Importing Matplot Library in order to visualise the data in barplot

import matplotlib.pyplot as plt

cols = 5
rows = 5
num_cols = df.select_dtypes(exclude='object').columns
fig = plt.figure( figsize=(cols*5, rows*5))
for i, col in enumerate(num_cols):
#for i in num_cols:
    ax=fig.add_subplot(rows,cols,i+1)

    sns.histplot(x = df[col], ax = ax)

fig.tight_layout()
plt.show()
```



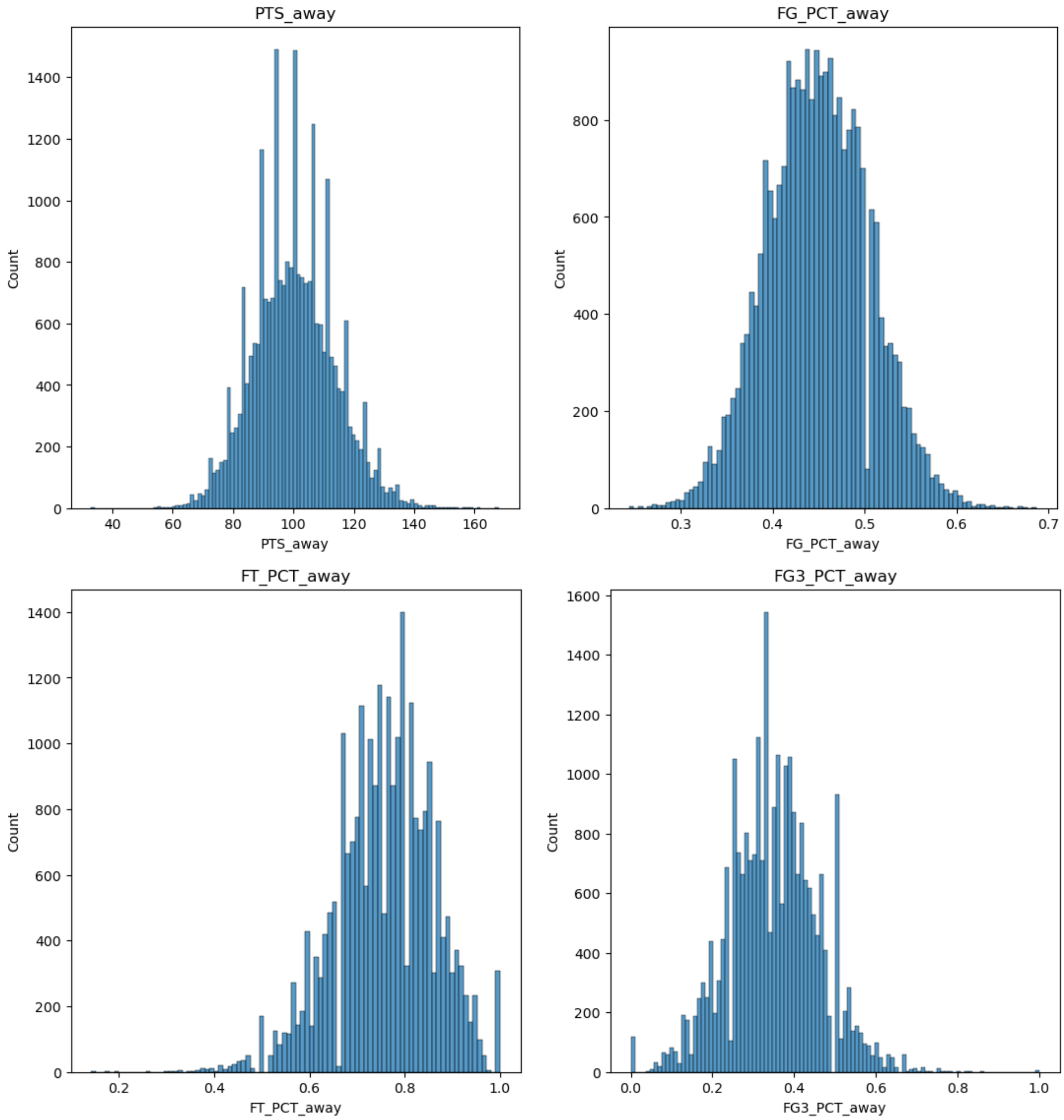
Histplot for given columns:

```
df[['PTS_away','FG_PCT_away','FT_PCT_away','FG3_PCT_away']].describe()
```

	PTS_away	FG_PCT_away	FT_PCT_away	FG3_PCT_away
count	25697.000000	25697.000000	25697.000000	25697.000000
mean	100.294120	0.449265	0.758082	0.349413
std	13.343016	0.055528	0.103418	0.110194
min	33.000000	0.244000	0.143000	0.000000
25%	91.000000	0.412000	0.692000	0.278000
50%	100.000000	0.448000	0.765000	0.350000
75%	109.000000	0.487000	0.833000	0.420000
max	168.000000	0.687000	1.000000	1.000000

Ploting the Histplot for the above columns

```
plt.figure(figsize=(20,14))
plt.title("Histplot")
plt.subplot(231)
plt.title("PTS_away")
sns.histplot(df["PTS_away"])
plt.subplot(232)
plt.title("FG_PCT_away")
sns.histplot(df["FG_PCT_away"])
plt.figure(figsize=(20,14))
plt.subplot(231)
plt.title("FT_PCT_away")
sns.histplot(df["FT_PCT_away"])
plt.subplot(232)
plt.title("FG3_PCT_away")
sns.histplot(df["FG3_PCT_away"])
plt.show()
```

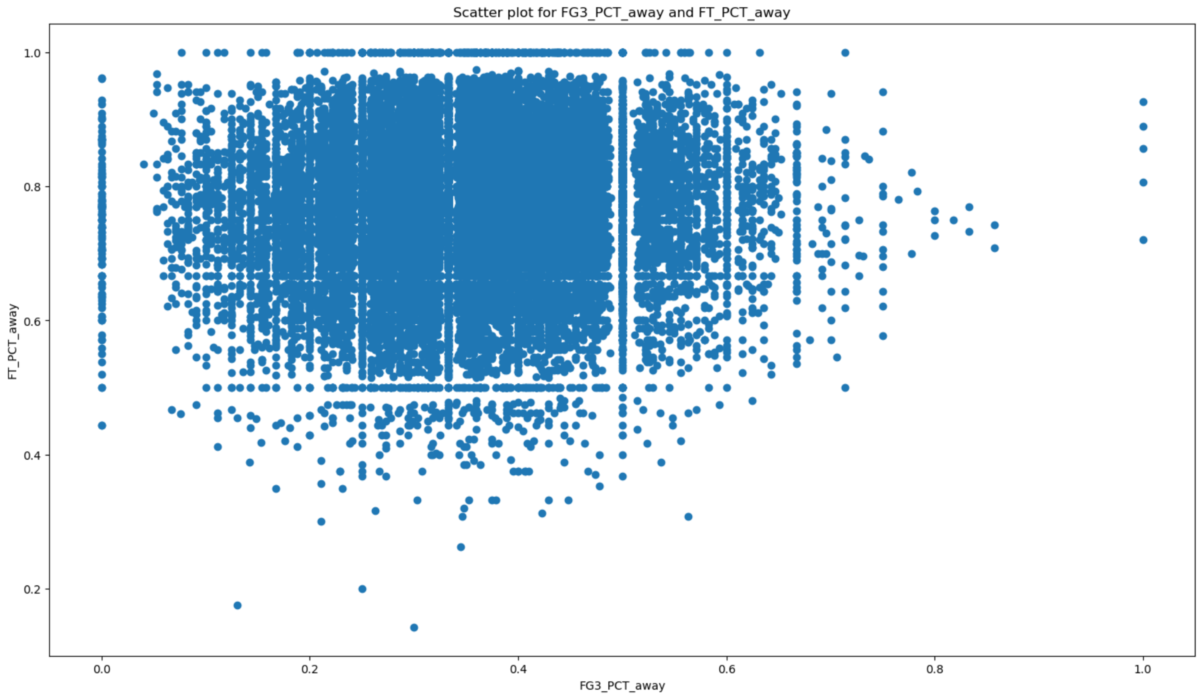


Scatter plot for given columns:

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize = (18,10))
plt.title("Scatter plot for FG3_PCT_away and FT_PCT_away")
ax.scatter(df['FG3_PCT_away'], df['FT_PCT_away'])

# x-axis label
ax.set_xlabel('FG3_PCT_away')

# y-axis label
ax.set_ylabel('FT_PCT_away')
plt.show()
```



Univariate Analysis:

```
df["HOME_TEAM_WINS"].value_counts().sort_values()
```

```
0    10542
1    15155
Name: HOME_TEAM_WINS, dtype: int64
```

```
df.describe(include='object')
```

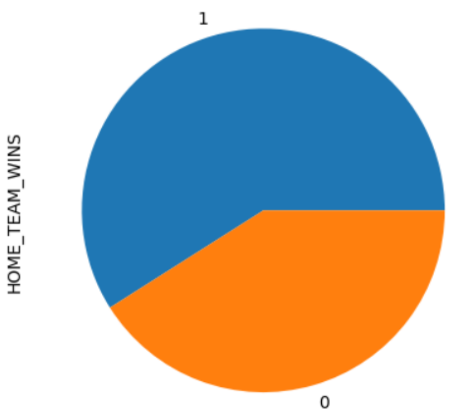
	GAME_DATE_EST	GAME_STATUS_TEXT
count	25697	25697
unique	4133	1
top	2020-12-23	Final
freq	16	25697

```
df['HOME_TEAM_WINS'].value_counts()
```

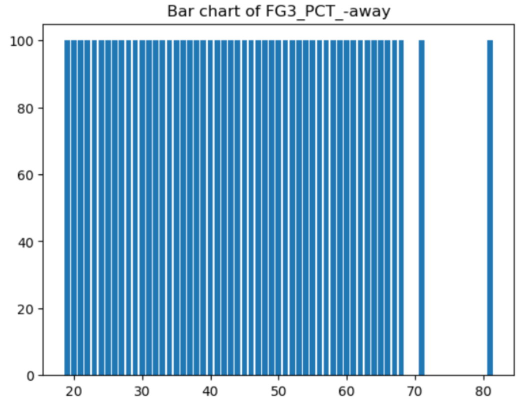
```
1    15155
0    10542
Name: HOME_TEAM_WINS, dtype: int64
```

```
df['HOME_TEAM_WINS'].value_counts().plot.pie()
```

<AxesSubplot:ylabel='HOME_TEAM_WINS'>

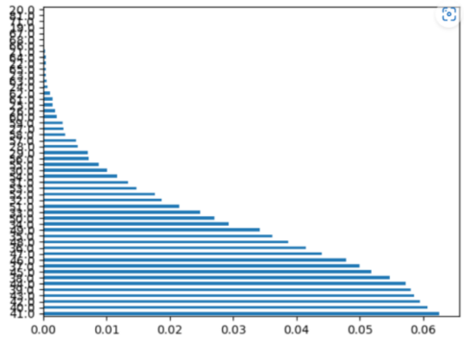


```
plt.bar(df["REB_away"],height=100)
plt.title("Bar chart of FG3_PCT_-away")
plt.figure(figsize=(19,90))
plt.show()
```

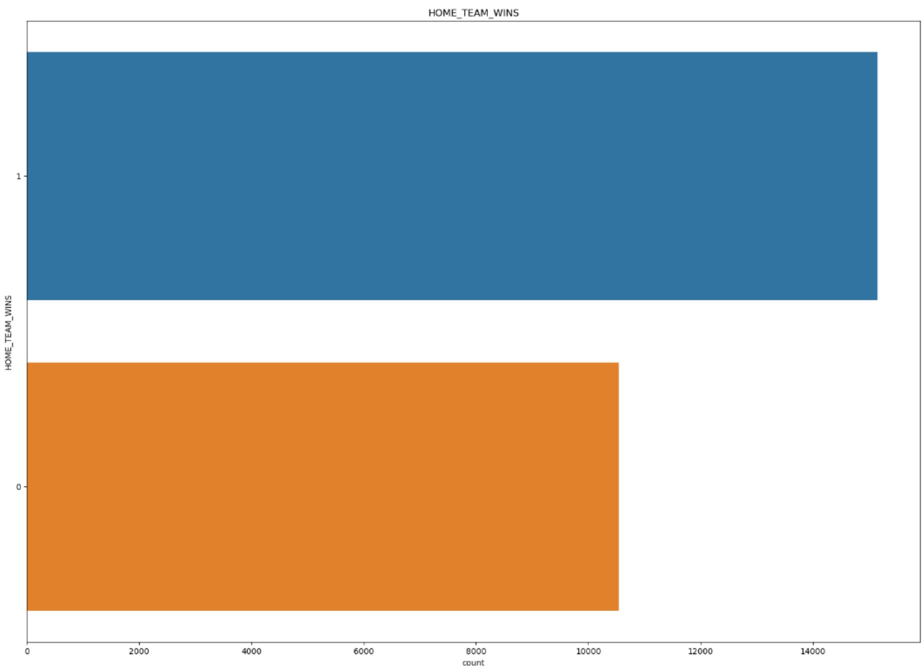



```
df['REB_away'].value_counts(normalize=True).plot.barh()
```

<AxesSubplot:>

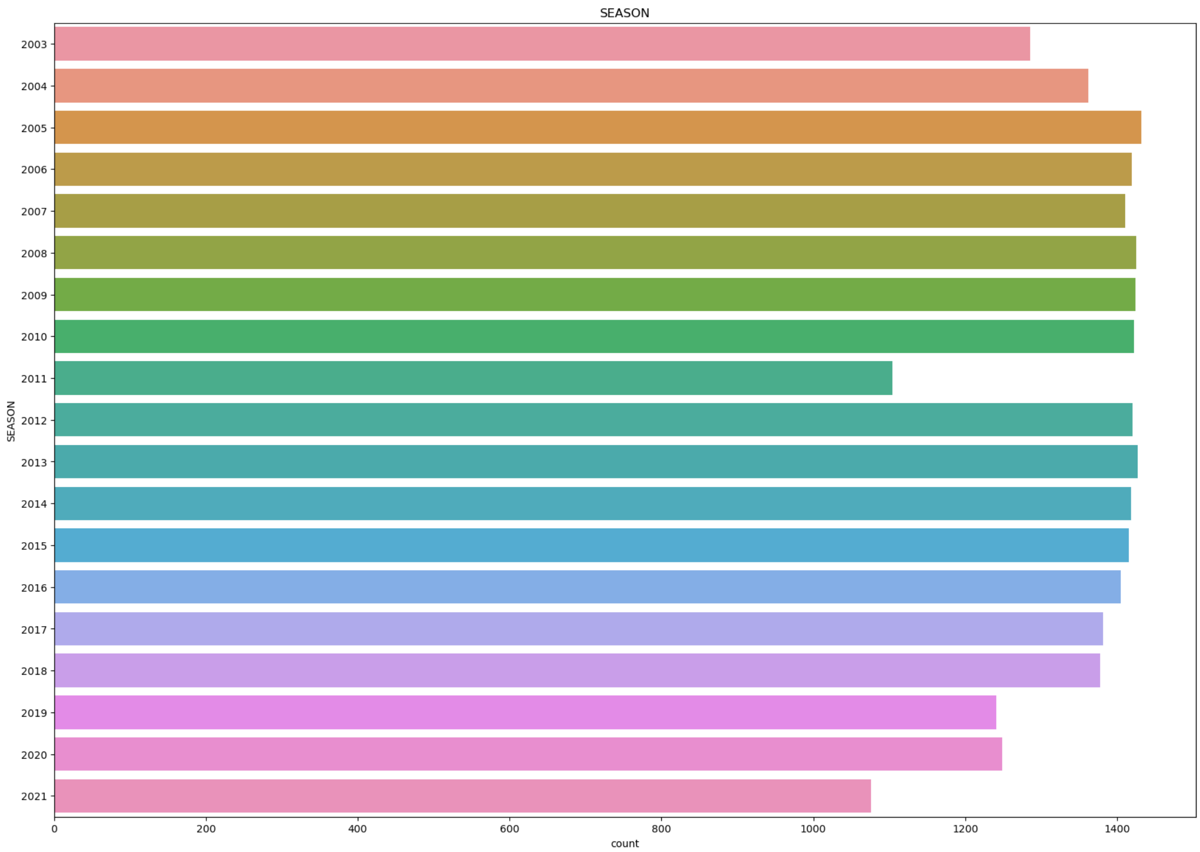


```
plt.figure(figsize=(20,14))
plt.title("HOME_TEAM_WINS")
sns.countplot(y="HOME_TEAM_WINS",data=df,order=df["HOME_TEAM_WINS"].value_counts().index)
plt.show()
```



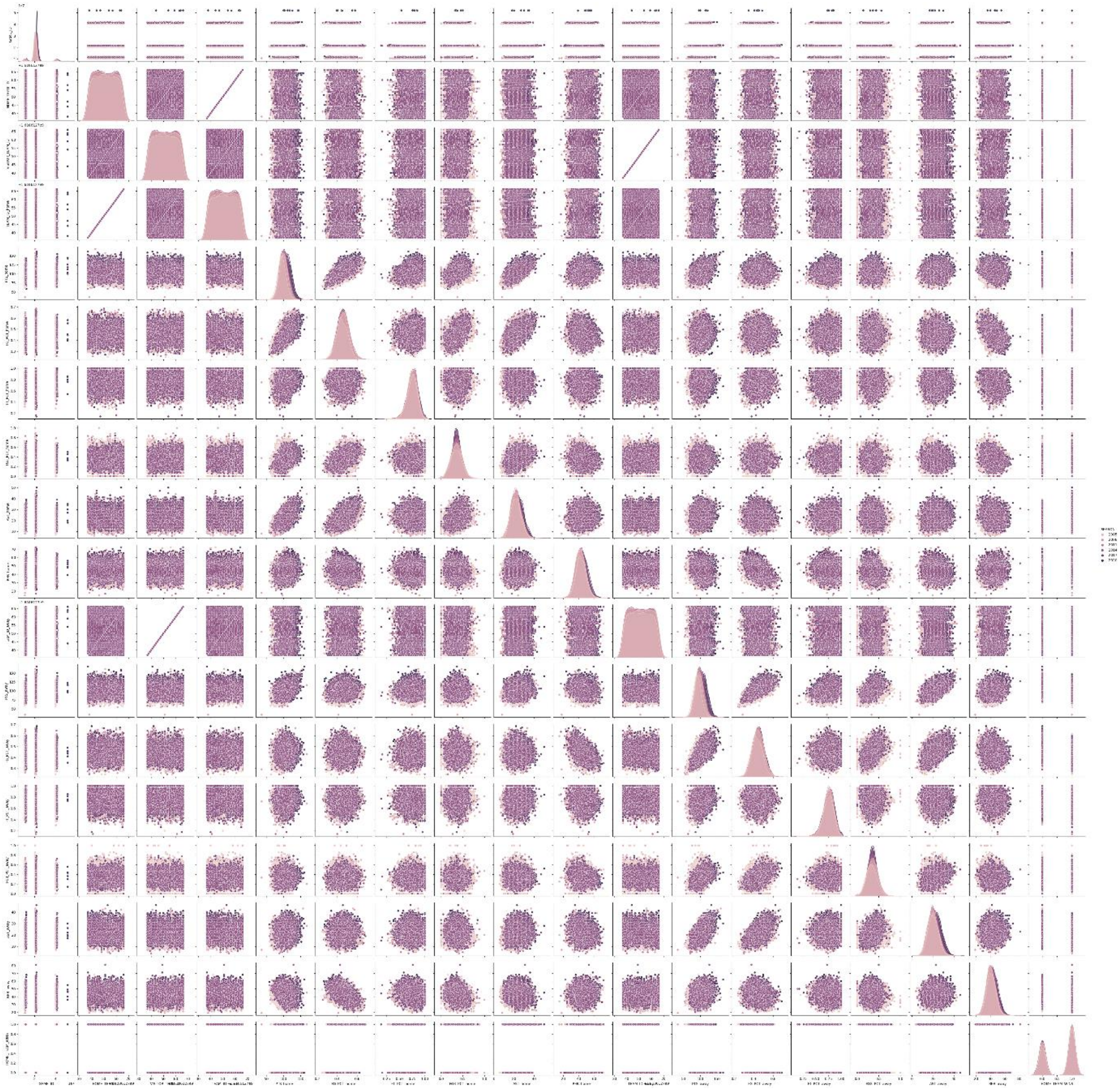
Bivariate Analysis:

Bivariate analysis can be defined as the **analysis of bivariate data**. It is one of the simplest forms of statistical analysis, which is used to find out if there is a relationship between two sets of values.



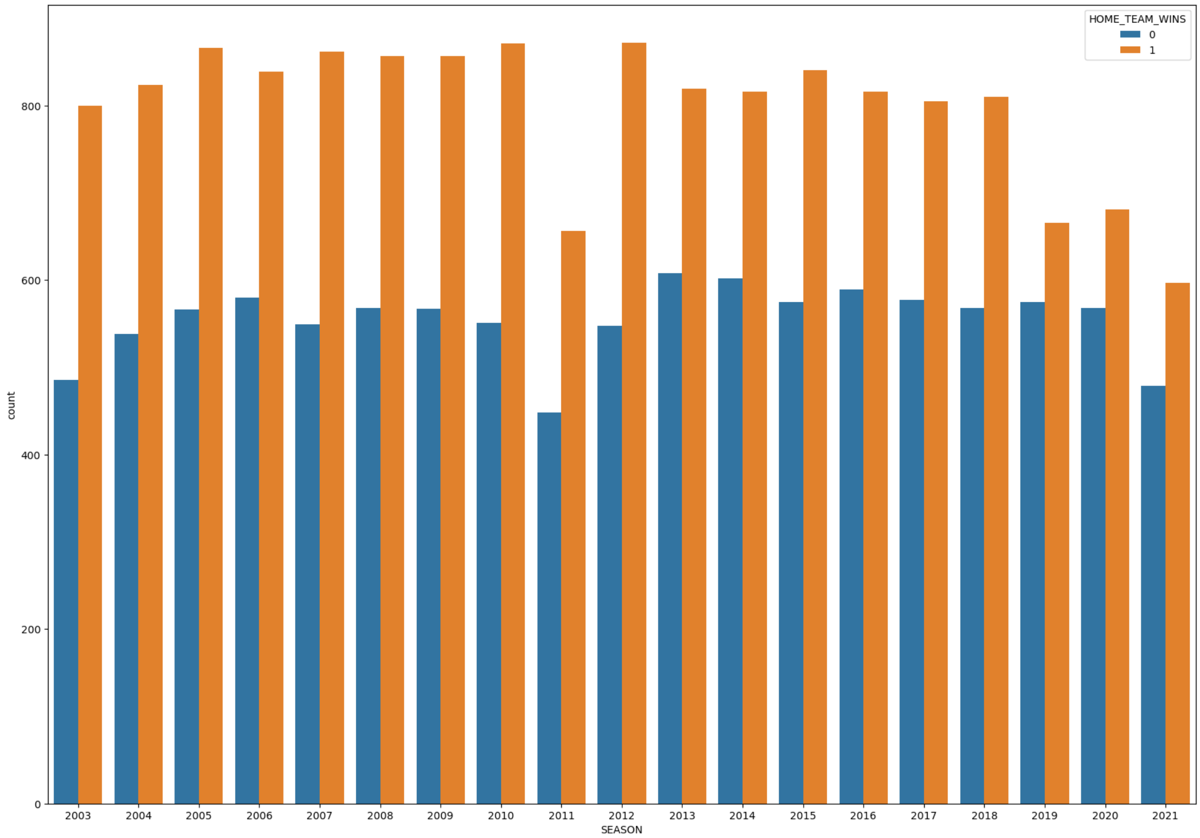
The above countplot is a seasons plot for every year which the football held.

The below is plot is pairplot for the column seasons



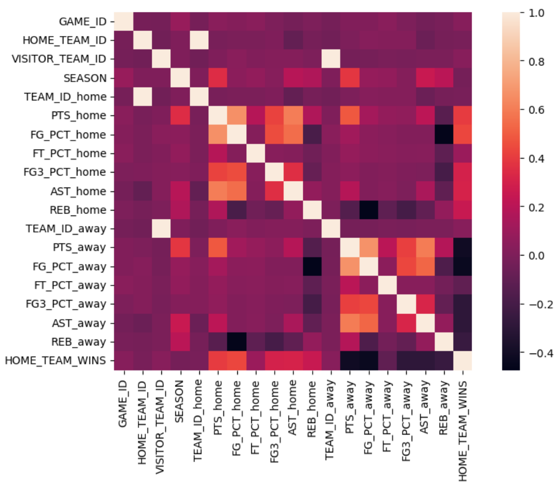
Countplot for the given columns:

```
plt.figure(figsize=(20,14))
sns.countplot(x="SEASON",hue="HOME_TEAM_WINS",data=df,order=df["SEASON"].value_counts().index.sort_values())
plt.show()
```



Heatmap graph for the dataset:

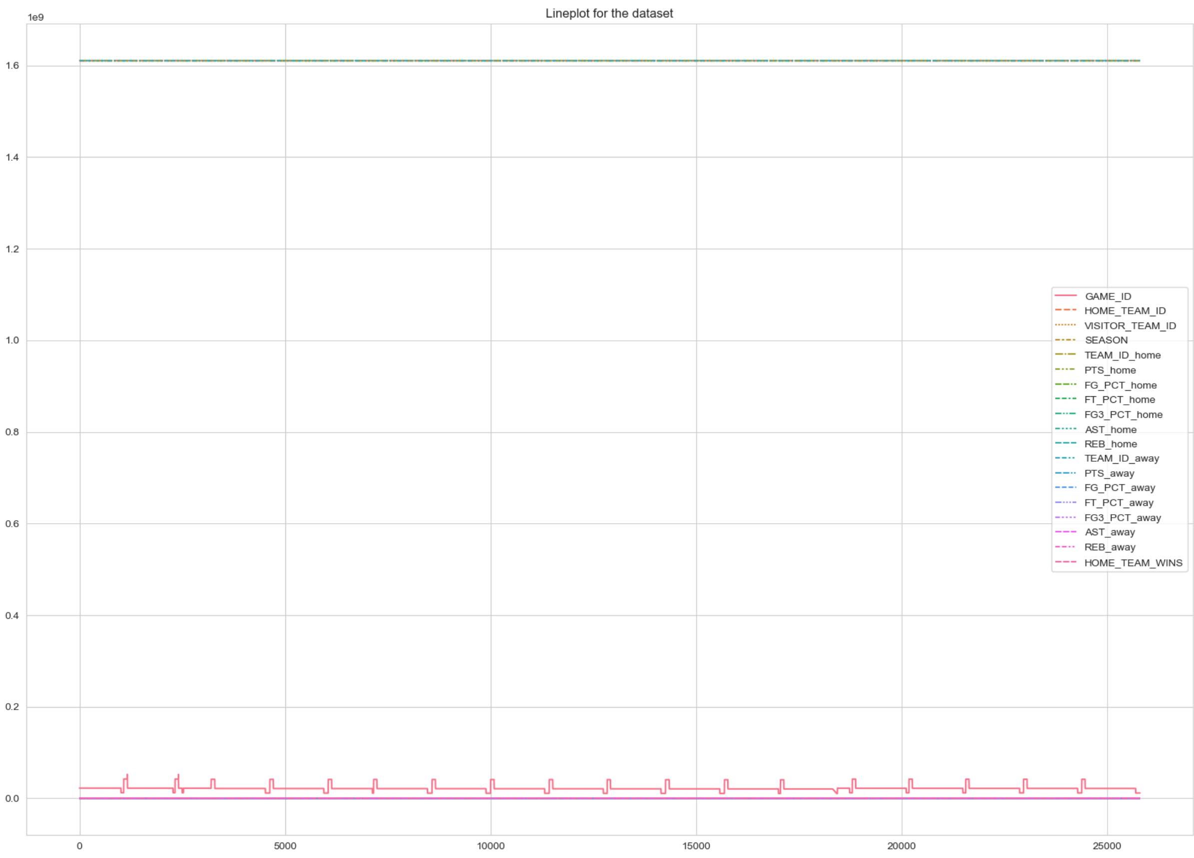
```
# Shows the heatmap graph
corr_df = df.corr()
plt.figure(figsize=(10,6))
sns.heatmap(corr_df,square=True)
plt.show()
```



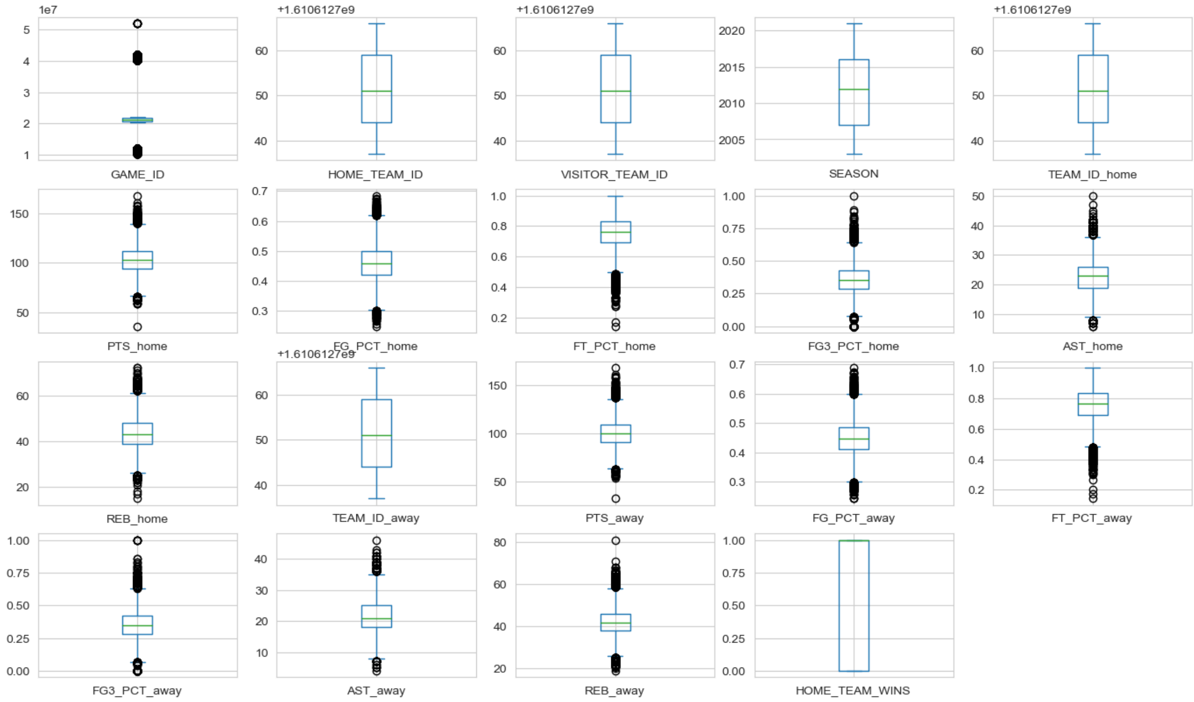
Multivariate Analysis:

Multivariate analysis is **a set of techniques used for analysis of data sets that contain more than one variable**, and the techniques are especially valuable when working with correlated variables. The techniques provide an empirical method for information extraction, regression, or classification; some of these techniques have been developed quite recently because they require the computational capacity of modern computers.

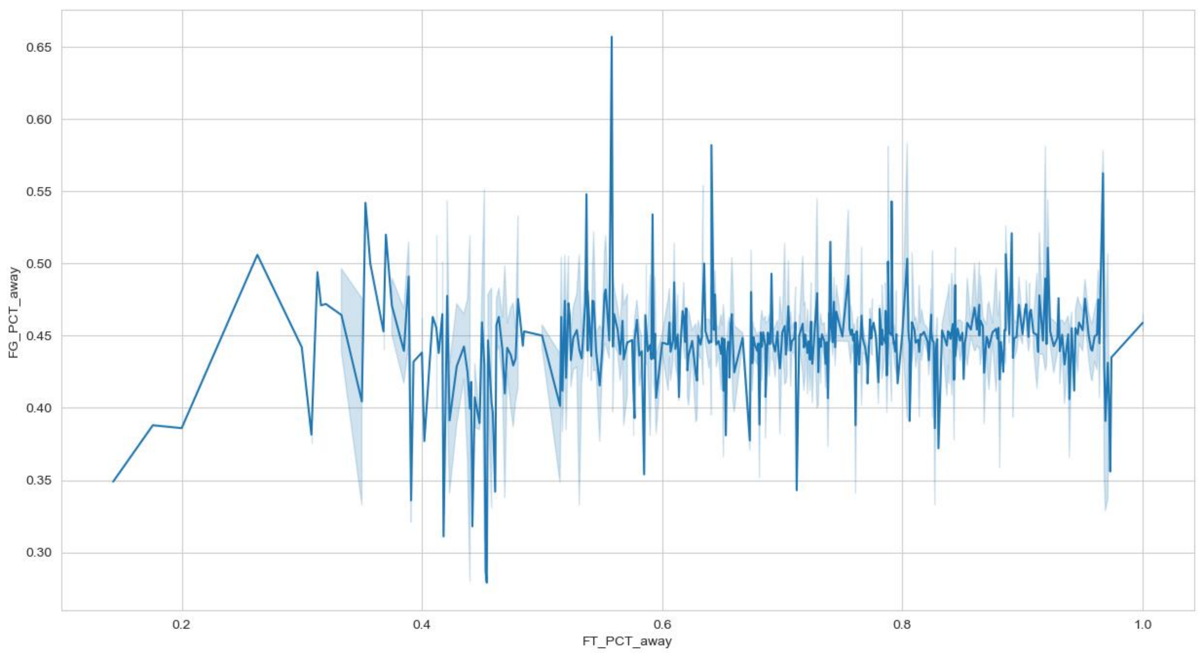
The above lineplot for the whole dataset indicate that the constant values and different values but no include the float values in the dataset there are more float values.so,we can't find the countious lines in this lineplot



Here,is the boxplot for the whole dataset

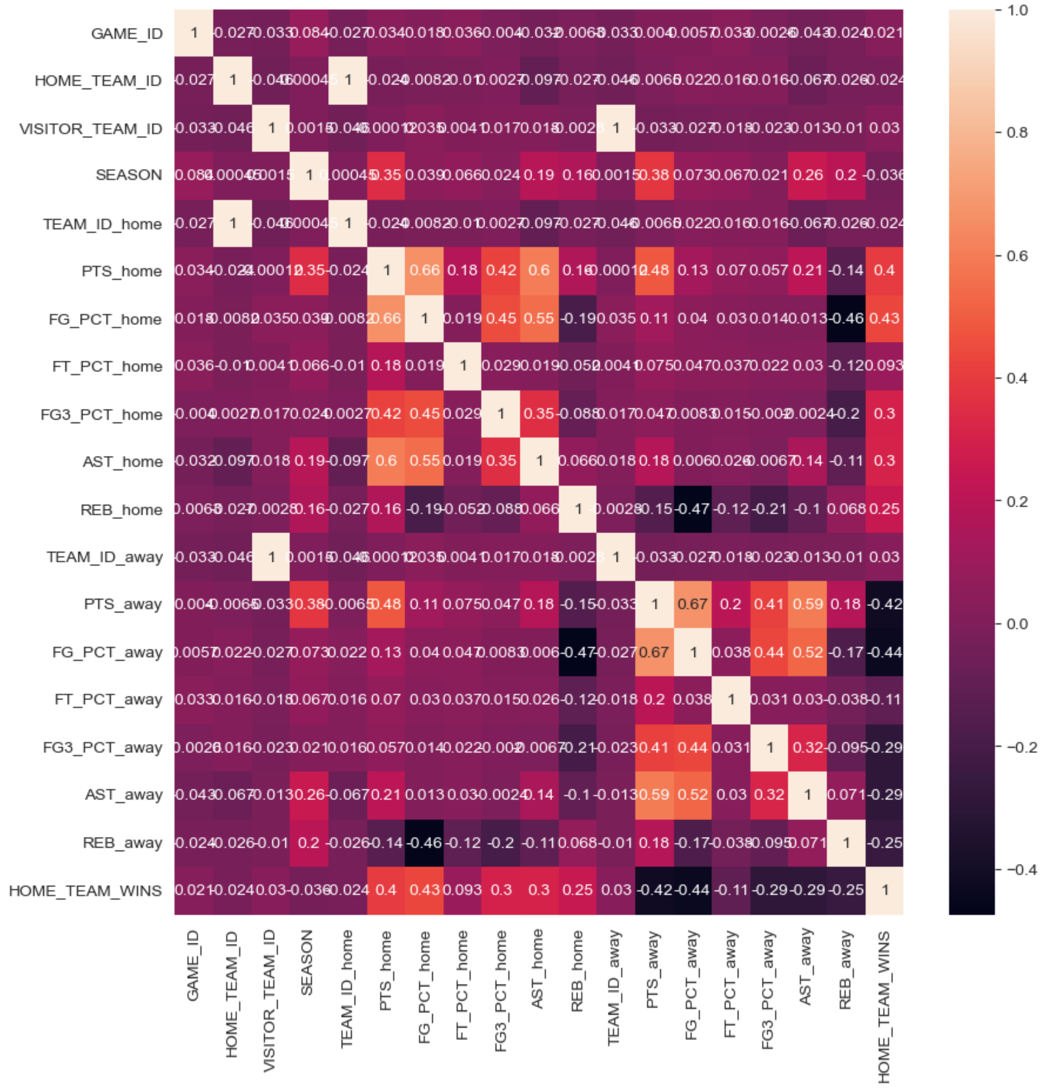


This is the lineplot for the two columns:



<u>CORRELATION using Heatmap</u>:

Correlation heatmaps are a type of plot that visualize the strength of relationships between numerical variables. Correlation plots are used to understand which variables are related to each other and the strength of this relationship. A correlation plot typically contains several numerical variables, with each variable represented by a column. The rows represent the relationship between each pair of variables. The values in the cells indicate the strength of the relationship, with positive values indicating a positive relationship and negative values indicating a negative relationship. In addition, correlation plots can be used to identify outliers and to detect linear and nonlinear relationships.



Conclusions after analyzing Heatmap:

• There are several variables that have no correlation and whose correlation value is near 0.

• A correlation heatmap is a graphical representation of a correlation matrix representing the correlation between different variables.

• The value of correlation can take any value from -1 to 1.

FINDING MEAN MEDIAN AND MODE SKLEARN METHOD:

```
from sklearn.impute import SimpleImputer
```

```
impo = SimpleImputer(strategy='mean')
x = df[['FG3_PCT_away']]
X = impo.fit_transform(x)
print(X)
```

```
[[0.357]
 [0.208]
 [0.389]
 ...
 [0.5  ]
 [0.385]
 [0.438]]
```

```
impo = SimpleImputer(strategy='median')
x = df[['FG3_PCT_away']]
X = impo.fit_transform(x)
print(X)
```

```
[[0.357]
 [0.208]
 [0.389]
 ...
 [0.5  ]
 [0.385]
 [0.438]]
```

```
impo = SimpleImputer(strategy='most_frequent')
x = df[['FG3_PCT_away']]
X = impo.fit_transform(x)
print(X)
```

```
[[0.357]
 [0.208]
 [0.389]
 ...
 [0.5  ]
 [0.385]
 [0.438]]
```

Conclusion:

In this report, we discussed the different methods used for data analysis, namely the Univariate, Bivariate, and Multivariate analysis techniques. These are classified based on the number of variables involved in the analysis. Under each analysis, we discussed some methods used to analyze the data and implemented them in python under each analysis.

Data is Categorized based on its datatype, and accordingly the data is visualized in several forms like Histplot, Boxplot, Kernel Density Plot, Violin plot, Bar plot, Pie chart, Pair plot, Scatterplot, Strip plot.

Choosing the correct way for the analysis depends on the type of data we are handling and the number of variables involved in the analysis. We also have done the statistical analysis we found out mean,median,mode,standard deviation, min and max value.

Reference:

Dataset is from Kaggle

Link: NBA games data | Kaggle

My Github

Link: swethak2/NBA-games-data (github.com)